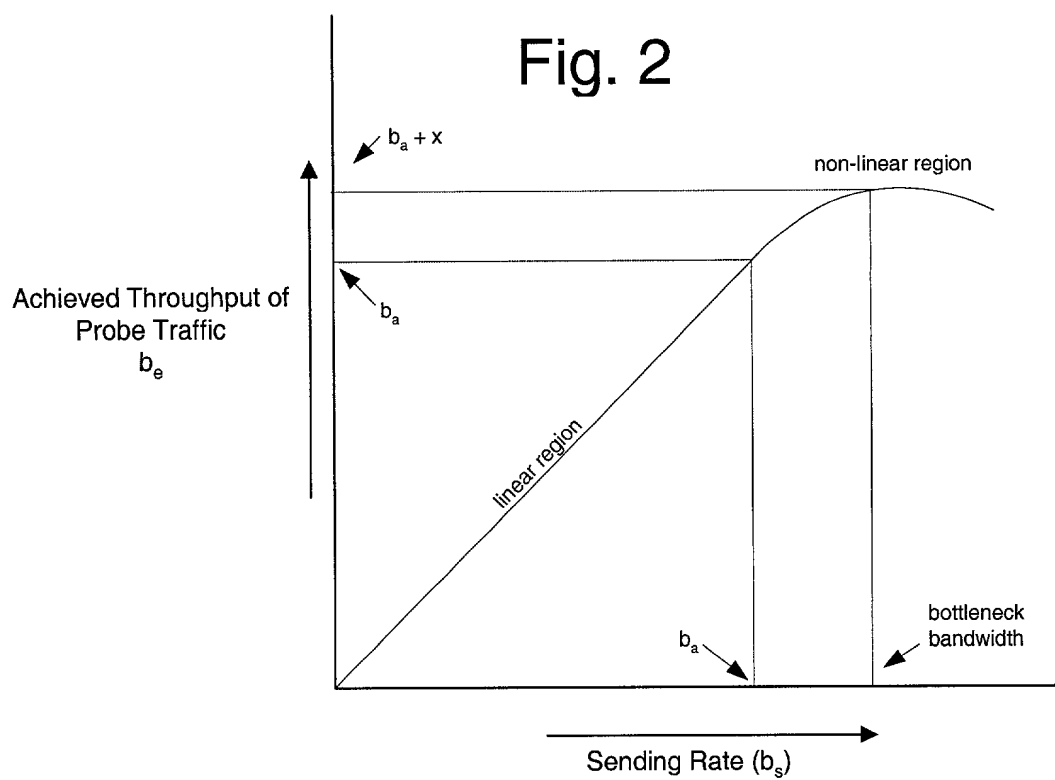


Fig. 1



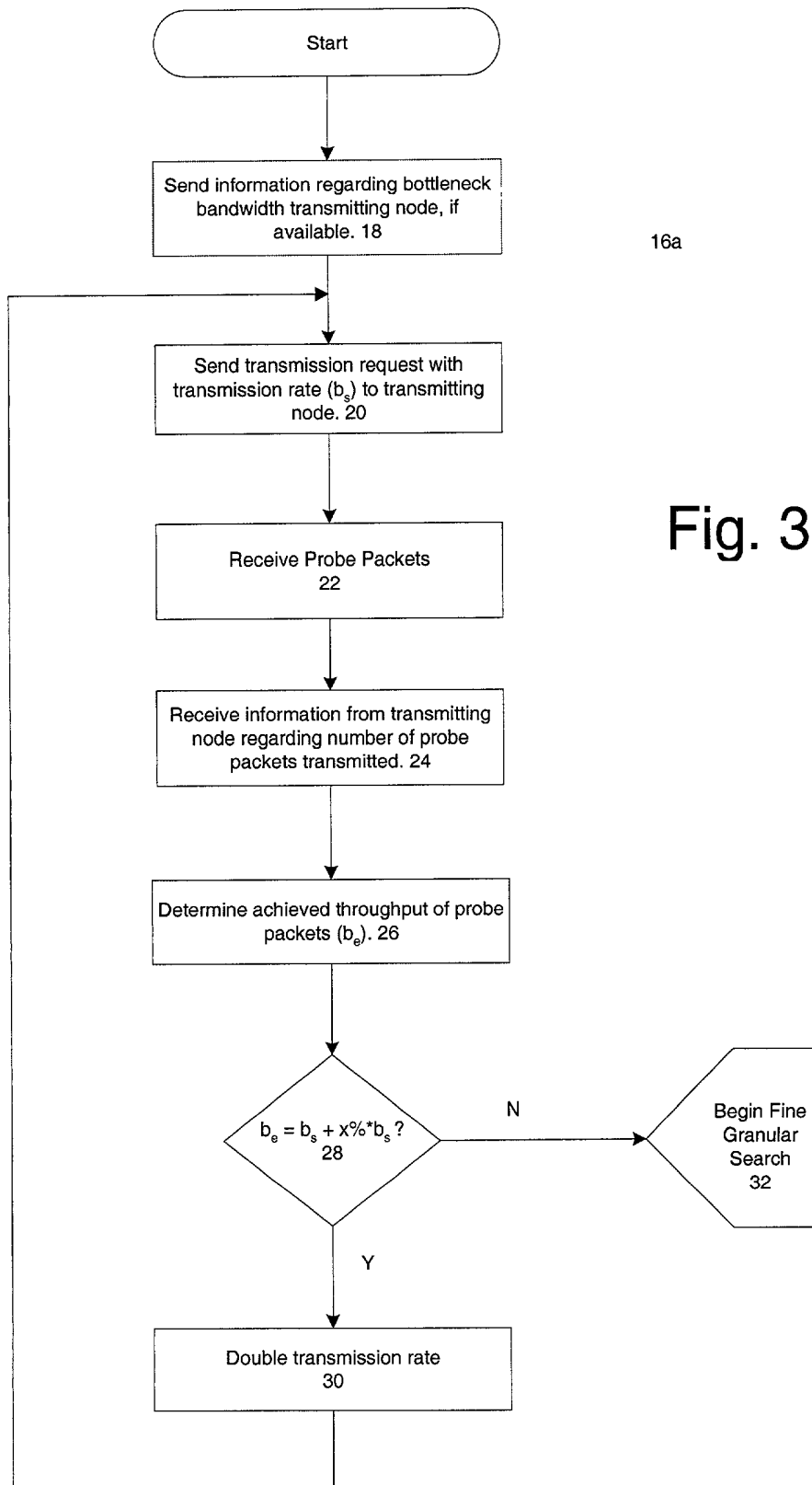
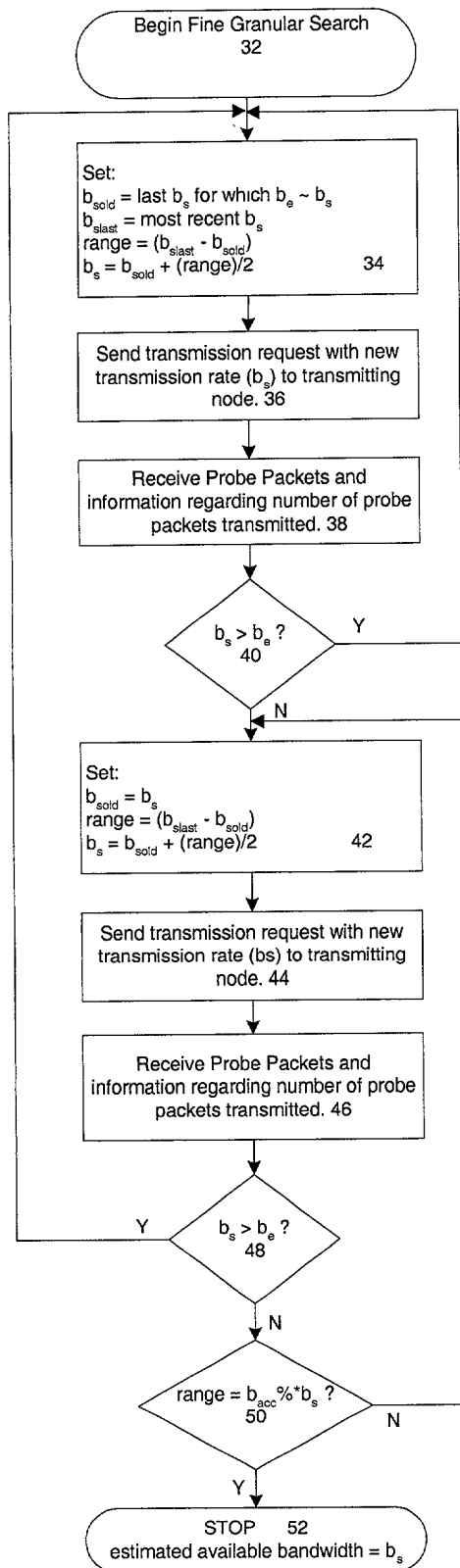


Fig. 3



16b

Fig. 4

Fig. 5

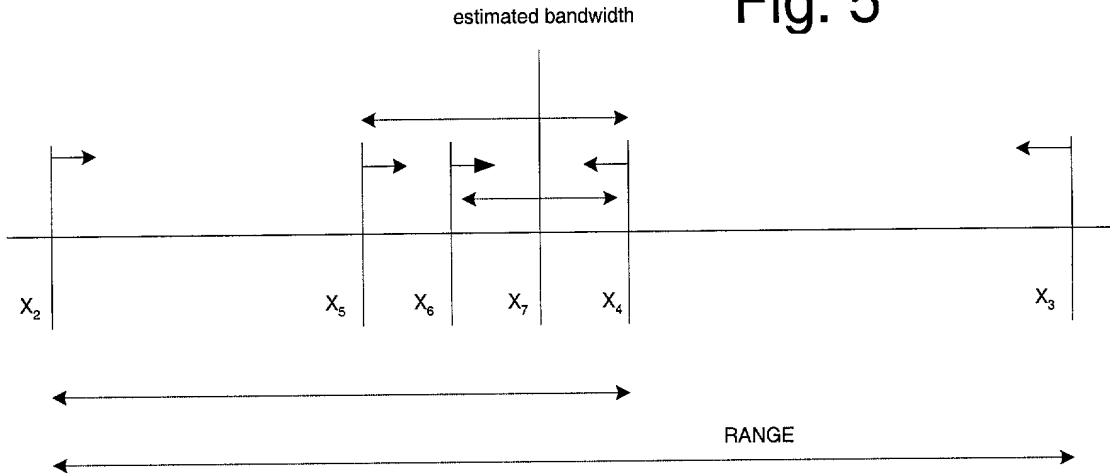
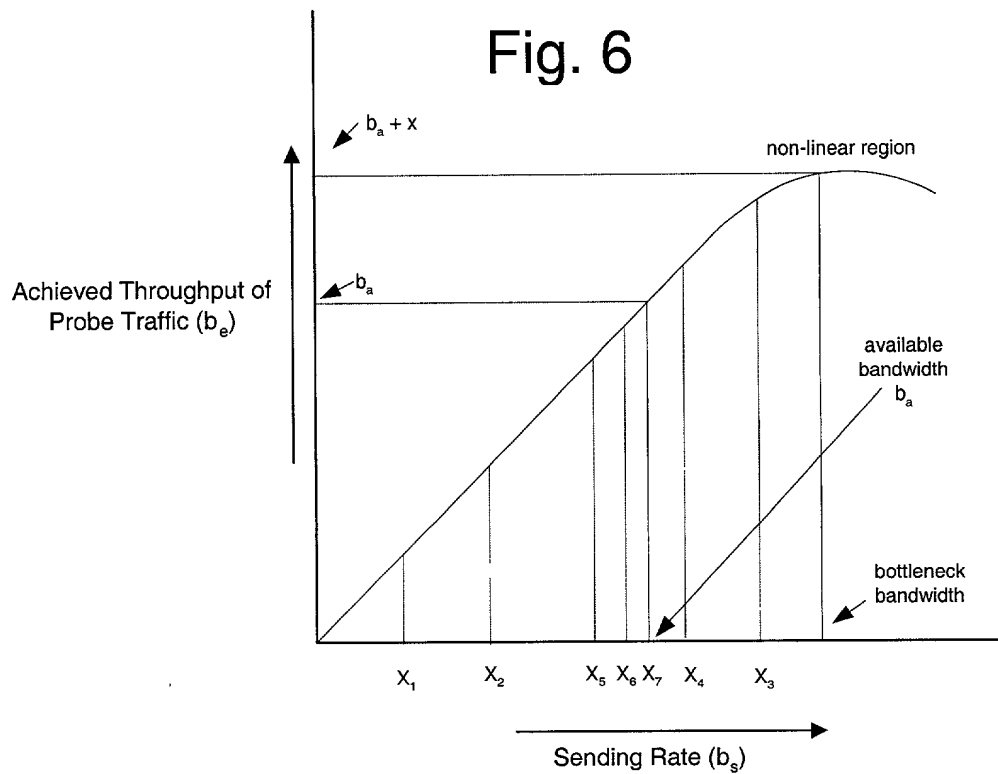


Fig. 6



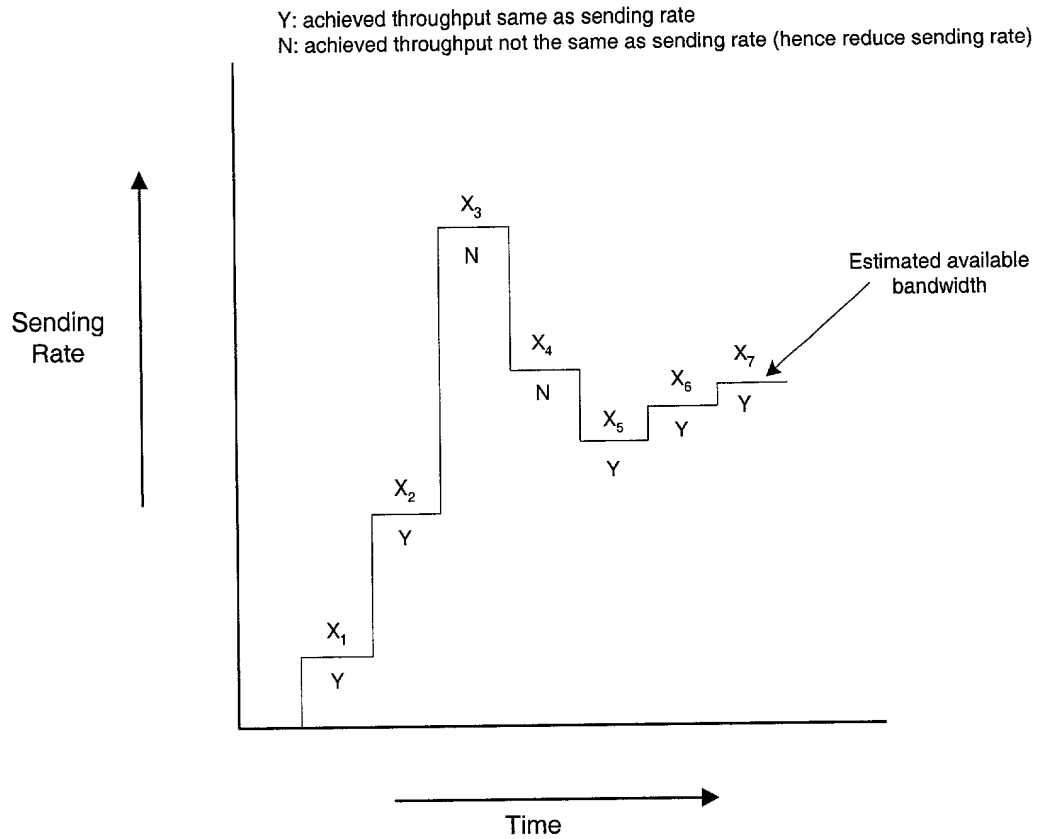


Fig. 7

---

```

01 Procedure Receiver.Initialize(sender)
02   open up TCP socket with the sender
03   open up UDP socket
04   rnd_no ← 0
  
```

---

Fig. 8

---

**Procedure exp\_growth()**

global constants : fraction\_start, increase\_factor, fixed\_minimum\_rate

---

```
01 when Sender initiates connection
02   receiver acks and asks sender for bottleneck bandwidth
03   wait for bottleneck bandwidth information from sender
04   receives bottleneck bandwidth information
05 if (bottleneck bandwidth available)
06   start_sending_rate = fraction_start*bottleneck bandwidth
07   increase_factor = 2
08 else
09   start_sending_rate = fixed_minimum_rate
10   increase_factor = 10
11 control_info.sending_rate = start_sending_rate
12 control_info.time = random(min_duration, max_duration)
13 control_info.rnd = ++rnd
14 send control_info to the sender
15 listen to probe packets on UDP socket and control packets on TCP socket
16 if (received probe_packet)
17   if (probe_packet is first packet of rwnd)
18     t_first ← time when received
19     seq_first ← sequence number of packet
20   if (probe_packet is last packet of rwnd)
21     t_last ← time when received
22     seq_last ← sequence number of packet
23   if (packet.seq_no < sequence number of last packet of same rwnd)
24     packet_reordering ← TRUE
25   if (received control_packet)
26     total_sent_bytes = control_packet.total_bytes
27   if (seq_last < pkid AND total_sent_bytes > total_rcvd_bytes)
28     packet_loss ← TRUE
29   if (packet_loss || packet_reordering)
30     try one more round if not already tried for 3 rounds with same sending rate
31     control_info.sending_rate = last_sending_rate
32     control_info.rnd = ++rnd
33     control_info.time = random(min_duration, max_duration)
34   go to 13
```

could next page

---

**Fig. 9a**

---

Procedure *exp\_growth()* contd...

---

```
30 else
31   achieved_throughput = total_sent_bytes / (t_end - t_start)
32   if ((achieved_throughput - sent_rate) / sent_rate) * 100 < ε
33     control_info.sending_rate = increase_factor * last_sending_rate
34     control_info.rnd = ++rnd
35     control_info.time = random(min_duration, max_duration)
36     min_range = control_info.sending_rate
37     if (control_info.sending_rate > bottleneck_bandwidth)
38       max_range ← bottleneck_bandwidth
39     else
40       go to L3
41     return min_range, max_range
42   else
43     max_range ← control_info.sending_rate
44     return min_range, max_range
```

---

**Fig. 9b**

global constants : fraction\_start, increase\_factor, fixed\_minimum\_rate  
min\_duration, max\_duration, acceptable\_fraction  
no\_bw\_acceptable\_fraction

```

01 range ← max_range - min_range
02 control_info.sending_rate ← min_range + range/2
03 control_info.time = random(min_duration, max_duration)
04 control_info.rnd = ++rnd
05 send control_info to the sender
06 listen to probe packets on UDP socket and control packets on TCP socket
07 if (received probe_packet)
08     if (probe_packet is first packet of rnd)
09         t_first ← time when received
10         seq_first ← sequence number of packet
11     if (probe_packet is last packet of rnd)
12         t_last ← time when received
13         seq_last ← sequence number of packet
14     if (packet.seq_no < sequence number of last packet of same rnd)
15         packet_reordering ← TRUE
16     if (received control_packet)
17         total_sent_bytes = control_packet.total_bytes
18     if (seq_last < peerID AND total_sent_bytes > total_recv_bytes)
19         packet_loss ← TRUE
20 if (packet_loss || packet_reordering)
21     try one more round if not already tried for 3 rounds with same sending rate
22     control_info.sending_rate = last_sending_rate
23     control_info.rnd = ++rnd
24     control_info.time = random(min_duration, max_duration)
25     go to 13
26 else
27     achieved_throughput = total_sent_bytes / (t_last - t_first)
28     if ((achieved_throughput - sent_rate) / sent_rate) * 100 < ε
29         min_range ← current_sending_rate

```

could next page

**Fig. 10a**



---

Procedure *fine\_granular\_search()* contd...

---

```
31   control_info.sending_rate = min_range + range/2
32   range = max_range - min_range
33   if (bottleneck_bandwidth_information_available)
34       if (range/bottleneck_bandwidth < acceptable_fraction)
35           return min_range + range/2
36       else
37           control_info.time = random(min_duration, max_duration)
38           control_info.md = ++rnd
39           go to Line 5
40   else
41       if (range/min_range_bandwidth < no_bw_acceptable_fraction)
42           return min_range + range/2
43       else
44           control_info.time = random(min_duration, max_duration)
45           control_info.md = ++rnd
46           go to Line 5
47   else
48       max_range = lastcurrent_sending_rate
49       range = max_range - min_range
50       control_info.sending_rate = min_range + range/2
```

---

Fig. 10b

---

**Procedure sender(receiver)**

global constants : fraction\_start, increase\_factor, fixed\_minimum\_rate  
 min\_duration, max\_duration, acceptable\_fraction  
 no\_bw\_acceptable\_fraction, pkt\_size

---

```

01 Initiate tcp connection with the receiver
02 After Receiver acks
03   open UDP socket for sending probe packets
04   sends receiver information about bottleneck bandwidth if available
05   when control_info is received over the tcp channel, terminates if
       received TERMINATION signal, prints the available bandwidth information
       provided by the sender
06   sending_rate = control_info.sending_rate
07   tot_time = control_info.time
08   rnd_no = control_info.md
09   avg_pkt_rate = sending_rate/packet_size
10   pkt_id = 0
10  starting_time = PRESENT.TIME
11  prepare a pkt
12  pkt.pkt_id = pkt_id
13  pkt.md = rnd_no
14  pkt.size = pkt_size
15  send the packet
16  if (PRESENT.TIME - starting_time) < tot_time
17    wait(1/avg_pkt_rate)
18  else
19    control_info.sending_rate = sending_rate
20    control_info.time = PRESENT.TIME - starting_time
21    control_info.md = rnd_no
22    control_info.last_id = pkt_id
    
```

---

**Fig. 11**